User Centric Hierarchical Classification and Associated Evaluation Measures for Document Retrieval

Korinna Bade and Andreas Nürnberger
Fakultät für Informatik, Institut für Wissens- und Sprachverarbeitung,
Otto-von-Guericke-Universität Magdeburg, D-39106 Magdeburg, Germany
{kbade,nuernb}@iws.cs.uni-magdeburg.de

Abstract

The classification of documents/objects into hierarchical structures is a problem of increasing importance, e.g. considering the growing use of ontologies or keyword hierarchies in many web-based information systems. Therefore, it is not surprising that it is a field of ongoing research. Here, we propose an approach that takes hierarchy information in the classification process into account by utilizing the user behavior in the information seeking process. In contrast to other methods, the hierarchy information is used independently of the classifier rather than integrating it directly. This enables the use of arbitrary standard classification methods. Furthermore, we discuss how hierarchical classification can be evaluated appropriately, especially by considering the usefulness of the classification for a user. We present our algorithm and evaluate it on two datasets of web pages using Naïve Bayes and SVM as baseline classifiers. Significant improvements over this baseline could be found with all performance measures.

1 Introduction

Classifying objects into hierarchical structures is a task, which is needed in a growing number of applications, for example in maintaining ontologies or keyword hierarchies in web-based systems. This includes, e.g., classifying web pages into a web directory or a hierarchy that was defined by groups of users. As an example of such groups we mention social web communities such as, e.g., the open directory project [DMOZ, 2006]. Despite the possibility of creating such hierarchies manually and assigning documents by hand, automatic classification and hierarchy extension would be beneficial for many domains as the number of documents to classify has become huge.

Hierarchical structures are used to help locating information of value to a user. When searching for information in a hierarchy, the user has some idea about the topic this information belongs to. This knowledge is used to browse the hierarchy labels in a top-down manner until a subclass is found, which is expected to contain the information. Once the user reaches the most specific class describing his information need, he starts scanning the documents. If none contains the information, he might also browse more general classes, depending on how much time he is willing to spend for his search. However, it is very unlikely that he would browse other specific classes in branches of the hierarchy that deal with different topics.

What does this search behavior imply for a classification method? Each object that is classified in a wrong subclass will most likely not be retrieved by the user. In contrast to this, each object that is classified into a class that is a generalization of the correct class might still be retrieved. However, too much generalization also hinders the user's search for information as it contradicts the idea of hierarchically structuring the data. The strongest generalization would be storing everything in the root folder, which is the same as having no structure at all.

Another problem during document class assignment is that a specific class might not yet exist in the hierarchy for a certain document. Here, classification in one of the most specific classes would prevent the user from retrieving the document as he would not look so deep down the hierarchy. In this case, predicting a more general class is the only way of making retrieval possible.

In our opinion, it is important to capture this user behavior when designing and evaluating classification algorithms. Especially during evaluation this would lead to a more appropriate assessment on which algorithm better supports the user when looking for information. In the standard evaluation measures currently used, this is not considered. Therefore, we present a user centric adaptation of performance measures in Section 3. In Section 4, we then present an approach to hierarchical classification of documents. It aims on generalizing predictions that are uncertain to avoid misclassification and therefore allows more likely for retrieval. We then evaluate our algorithm in Section 5 by using our proposed user centric evaluation measures. In the following section, we start by reviewing related work.

2 Related Work

Most of the related work for hierarchical classification deals with integrating the hierarchy information into the classification process by adapting existing methods or building new classifiers. The authors of [Cai and Hofmann, 2004] try to integrate hierarchy information directly into a support vector machine (SVM) classifier by integrating a hierarchical loss function, which is motivated by a document filtering setting. Their motivation is similar to ours. Different SVM classifiers for each hierarchy node are learned by the authors of [Sun and Lim, 2001] and [Dumais and Chen, 2000] to find a suitable node in the tree. However, an inner node can only be predicted, when data is assigned to it. This is not appropriate as such inner nodes also carry important meaning defined by its child nodes.

The authors of [McCallum *et al.*, 1998] applied shrinkage to the estimates of a Bayes classifier to improve the

probability estimates. They reported large improvements. As estimating class probabilities is one step of our algorithm, this method could be applied in future work to improve the estimates. In [Cesa-Bianchi *et al.*, 2004], an incremental algorithm with performance close to SVM and also a new loss function for evaluation is presented. The authors propose to learn linear threshold classifiers for each node to decide whether a document should be classified into the node or further down the hierarchy.

In [Choi and Peng, 2004], a greedy probabilistic hierarchical classifier is used to determine the most suitable path in the hierarchy from the root to a leaf. In a second step, another classifier is used to determine the best class along this path. The authors also suggest some criteria to create a new category. In [Granitzer and Auer, 2005], the performance of two Boosting algorithms, BoosTexter and CentroidBooster, is compared to the performance of support vector machines.

The influence of different training sets (with and without using hierarchy information) is examined in [Ceci and Malerba, 2003] using Naïve Bayes and centroid learning for different numbers of extracted features. The author of [Frommholz, 2001] adapts the determined weights for each category for a certain document in a post-processing step by integrating the weights of all other categories according to their proximity (the distance in the category tree/graph) to this category. However, he makes no differences concerning the relation between two nodes.

In summary, three main approaches to hierarchical classification can be distinguished. Firstly, the original training data can be reinterpreted in a pre-processing step, which is mostly done by assigning it not only to one class but also to parent and/or child classes in the hierarchy. This approach can be critical in our context, as a class associated with a class is not merely the union of the classes of the successor classes. More specific, a document may belong to an inner class but not to any of the more specific topics associated with the successor classes.

Secondly, the hierarchy could be used directly in the classifier design, which means developing completely new classification methods. Some examples of this approach have been mentioned above.

Our approach belongs to a third category, in which the class hierarchy is exploited in a post-processing step. More specifically, this step consists of reinterpreting basic probability assignments, which typically come from a standard (non-hierarchical) classifier by integrating further knowledge. In other words, the hierarchical structure of the problem is exploited here. As an advantage of this class of methods let us mention that it allows for using well-established standard classifiers in the first step (this advantage is of course shared by the first class of methods).

3 Performance Measures for Hierarchical Classification

To compare results between different algorithms, it is necessary to define appropriate performance measures. For standard (flat) classification, evaluation is mostly done by precision and recall [Hotho *et al.*, 2005]. The precision of a class c is the fraction of all documents retrieved for this class that are correctly retrieved (see (1)), while the recall of c is the fraction of all documents belonging to this class that are actually retrieved (see (2)). The combination of the two, the F-Score, is usually used to evaluate overall performance (see (3)). Furthermore, the accuracy is often determined, which gives the percentage of correctly classified documents (see (4)).

$$prec_c = \frac{|relevant_c \cap retrieved_c|}{|retrieved_c|} \quad (1)$$

$$rec_c = \frac{|relevant_c \cap retrieved_c|}{|relevant_c|} \quad (2)$$

$$F_c = \frac{2}{1/rec_c + 1/prec_c} \quad (3)$$

$$acc = \frac{|\{d \in D | \text{predClass}(d) = \text{class}(d)\}|}{|D|} \quad (4)$$

These measures treat all classes equally. There is just one correct class and all others are wrong. However, as we already argued, in hierarchical classification, not all "wrong" classifications are equally "bad". Therefore, we propose a user oriented adaptation to the standard measures. Furthermore, collecting information about the types of misclassification can give more insight into the quality of the post-processing step.

3.1 Hierarchical Precision, Recall, and Accuracy

From our scenario at hand we derive the notion of the *retrieval path*, which starts at the correct class and goes up the hierarchy until the root class, i.e., the *retrieval path* rp_c associated with a class c contains all classes c_i from the hierarchy H which are either the class itself or a parent class thereof (denoted by $c_i \geq_H c$):

$$rp_c = \{c_i \in H | c_i \geq_H c\} \quad (5)$$

In Fig. 1, the retrieval path of class 4 is marked in bold as an example. Please note that the child classes of class 4 (here classes 6 and 7) do not belong to the retrieval path.

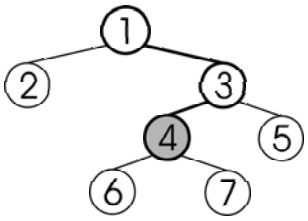

Figure 1: Example for a retrieval path

Furthermore, we denote by $\text{dist}_H(c_1, c_2)$ the number of edges on the path between c_1 and c_2 in the tree, e.g., $\text{dist}_H(4,4) = 0$ and $\text{dist}_H(1,4) = 2$.

As motivated by our retrieval scenario, classes, which belong to the retrieval path, are all beneficial to the user. However, the more the concept was generalized, the more unlikely it is that the user puts in the effort to retrieve this document. Therefore, the usefulness of a prediction along the retrieval path decreases on the way from the correct class to the root. To integrate this notion of prediction usefulness, we propose an adaptation to the standard measures, which uses gradual membership degrees of class matches.

As in the standard version, correctly classified documents are counted as a perfect match (with a value of 1) and documents that are not classified on the retrieval path are not counted at all (with a value of 0). However, documents that are classified into a more general concept along the retrieval path are now counted with a value between 0 and 1, depending on how far they are away in the class tree. These ideas can be expressed by a large number of

functions. Here, we propose the measure in (6) that shows the hierarchical similarity of a predicted class $\hat{c}$ to a class c.

$$\mathrm{sim_H}(\hat{c}, c) = \begin{cases} \exp(-\gamma \cdot \mathrm{dist_H}(\hat{c}, c)) & \text{if } \hat{c} \geq_H c \\ 0 & \text{else} \end{cases} \quad (6)$$

This function has a parameter γ that can be used to adapt the evaluation to certain types of users. More specifically, the "laziness" of the user can be modeled. The higher γ, the less would a user be willing to check documents in more general classes. Hence, it becomes more important to predict a class, which is close to the true class. In particular, it is worth considering the two extreme parameter configurations.

With $\gamma \to \infty$, every class except the correct one is assigned a value of 0. This configuration models an extremely lazy user who only explores a single class. In this configuration, the hierarchical similarity converges to the standard setting of evaluation measures. No distinction is made between incorrect predictions. Being on the retrieval path is no longer better than being aside.

The other extreme is obtained for $\gamma = 0$. Now, every class along the retrieval path has the same usefulness. Exploiting information about the hierarchical structure now becomes fairly easy. Predicting the root class by default is an optimal strategy and guarantees predicting the most useful class. Most users will most likely be found somewhere in between. For our evaluation, we therefore set γ to 0.5.

The $\mathrm{sim_H}$ function can be used to transform the crisp 0/1-assignment in the standard measures to gradual degrees. This is done in (7) - (9). Again, we want to mention that $\mathrm{sim_H}$ could be replaced by any other function computing values between 0 and 1 to express other class relationships.

$$prec_{h,c} = \frac{\sum_{d \in retrieved_c} \mathrm{sim_H}(c, \mathrm{class}(d))}{|retrieved_c|} \quad (7)$$

$$rec_{h,c} = \frac{\sum_{d \in relevant_c} \mathrm{sim_H}(\mathrm{predClass}(d), c)}{|relevant_c|} \quad (8)$$

$$acc_h = \frac{\sum_{d \in D} \mathrm{sim_H}(\mathrm{predClass}(d), \mathrm{class}(d))}{|D|} \quad (9)$$

Other researchers also proposed hierarchical performance measures, e.g. in [Sun and Lim, 2001]. However, their focus is more on evaluating the classifier performance itself, e.g. by taking category similarities or tree distances into account. Our focus is on the usefulness of the classification from a user's point of view, which is based on user behavior in the described application scenario.

3.2 Types of Misclassification

Besides the previously introduced performance measures, we are further interested in determining some statistics which allow for distinguishing different types of misclassification. These can help to get an idea about more specific behavior of the different classification methods. For example, the same accuracy values could be gained by different classifier behavior. One algorithm might have the tendency to predict for documents either the correct class or a class totally wrong, while another algorithm might prefer predicting many documents in some more general class along the retrieval path. We therefore also determined the following statistics:

(a) $\#n_c$ – number of predictions in the correct class

(b) $\overline{\#n_c}$ – predictions in a class that is a parent class of the correct class, i.e., predictions on the retrieval path but not the correct class itself

(c) $\overline{ml(n_c)}$ – average number of hierarchy levels between the correct class and the predicted class in case of (b)

(d) $\underline{\#n_c}$ – predictions of classes not on the retrieval path, i.e., not counted for $\#n_c$ and $\overline{\#n_c}$

(e) $\underline{ml(n_c)}$ – average number of hierarchy levels between the retrieval path and the predicted node in case of (d)

4 An Approach Based on Classification Uncertainty

In this section, we present an approach that uses the hierarchical class structure and further knowledge about the prediction process to avoid misclassification into classes that are not part of the retrieval path. This is done as a post-processing step, in which prediction probabilities are re-evaluated.

In our setting, we have given a class hierarchy H together with a set of training data D, whereby not every class must have training data assigned to it. Classes that have subclasses might be empty. However, the best prediction might be found in such a class. Furthermore, we have a classifier C that computes prediction probabilities $P_C(c|d)$ for each class c for a given document d. This can either be a flat (as in our experiments) or a hierarchical classifier. Keep in mind that a flat classifier would produce a probability of 0 for empty classes. In addition, the sum of the prediction probabilities for all classes does not need to sum up to 1 as the rest of the probability mass could be describing unknown classes. The goal of our approach is to find for each document a prediction that is either the correct class or another class on the retrieval path by being as specific as possible.

The basic idea of our approach is to traverse the class hierarchy top-down. At each class, the algorithm looks for the most suitable child class of the current class and decides whether it is still reasonable to descend to it or whether it is better to return the current class as the final prediction.

To be able to make this decision, we utilize two different kinds of information. The first one is based on the prediction probabilities $P_C(c|d)$ from the classifier. We integrate the hierarchical relationship between classes into these probabilities by propagating the maximum prediction probability up the hierarchy, i.e. for each class the following equation is computed:

$$P(c|d) = \max_{c' \leq_H c} P_C(c'|d) \quad (10)$$

Descending the class hierarchy by always choosing the subclass with the highest value $P(c|d)$ now produces an equal prediction to choosing the class with the highest classifier prediction $P_C(c|d)$. Please note that for this purpose for each original inner class of the hierarchy a virtual leaf class is created, which is a direct sub-class of the concerning class. The classifier predictions $P_C(c|d)$ of these classes are then associated to the virtual classes. This is needed as an internal representation but will not be visible to the user.

Second, we use the training data to derive further class information. In prior work [Bade and Nürnberger, 2005], we used class similarities. Here, we extract the probability of predicting a class correctly. We denote this as prediction accuracy of a class $P_A(c)$. By performing 10-fold cross-validation on the training data, we build a confusion matrix

M for the original classification. In a second step, we again utilize the hierarchy information to derive the prediction accuracy of each class by the following equation:

$$P_A(c) = \frac{\sum_{c_1,c_2 \leq_H c} M(c_1, c_2)}{\sum_{c_3 <_H c, c_4 \in H} M(c_3, c_4)} \quad (11)$$

In other words, we interpret each class as the combination of it with its subclasses. So for each class, the number of all documents is counted that belong to this class or a subclass and that are also predicted to belong to one of these classes. This number is set in relation to all predictions of this "group" of classes.

After having determined both probabilities for each class, we can formulate a criterion to decide whether further descend into the class hierarchy should be stopped or not. The hierarchy is traversed along the highest prediction probability. If the following criteria holds, the descend is stopped:

$$\sqrt{P(c_b|d) \cdot P(c_{sb}|d)} \cdot P_A(c_c)^t > P(c_b|d) \cdot P_A(c_b)^t \quad (12)$$

where c_c is the current class of interest and c_b and c_{sb} are the direct child classes with the highest and second highest prediction probability. t defines a threshold parameter, which can be used to tune the algorithm.

What does the above equation model? At both sides of the inequality, the class accuracy and the prediction probability of a class are combined, weighted by the parameter t. The main idea is to compare the combined value of the current class c_c (left side of inequality) with the combined value of the best child class c_b (right side of inequality). If the value decreases down the hierarchy, the algorithm stops.

However, the prediction probability in the current class is always equal to the prediction probability of the best child class due to our initialization procedure described in (10). Therefore, we decided to replace the prediction probability of the current class with the harmonic mean of the prediction probabilities of the two best child classes. The main idea is that this expresses the "purity" of the class assignment. If the two best class probabilities are almost the same, the classifier is rather undecided, which class is the best, producing an almost equal value for the prediction probability. And this is actually the kind of classification uncertainty, we want to detect and avoid by generalization. The parameter t can be used to tune how strong the influence of the class accuracy should be for this matter.

The complete algorithm of our approach to user centric hierarchical classification (UCHC) is summarized in Fig. 2.

UCHC(d, C, γ)
- For each $c_i \in H$:
  - Compute probability estimate $P_C(c_i|d)$ by C
- For each $c_i \in H$:
  - Compute $P(c_i|d) = \max_{c' \leq_H c_i} P_C(c'\,d)$
- Build M by running C on training data
- For each $c_i \in H$:
  - Compute $P_A(c_i) = \frac{\sum_{c_1,c_2 \leq_H c_i} M(c_1,c_2)}{\sum_{c_3 \leq_H c_i, c_4 \in H} M(c_3,c_4)}$
- $c_c = root(H)$
- While c_c has child classes:
  - Determine the two child classes c_b and c_{sb} of c_c that have the highest values for $P(c|d)$
  - If $\sqrt{P(c_b|d) \cdot P(c_{sb}|d)} \cdot P_A(c_c)^t > P(c_b|d) \cdot P_A(c_b)^t$
    - break
  - $c_c = c_b$
- Return c_c

Figure 2: Our User Centric Hierarchical Classification Algorithm (UCHC)

5 Evaluation

To evaluate our algorithm, we used two different, well-established flat classifiers over all classes containing training data as reference classifiers, a standard Naïve Bayes classifier and a SVM classifier (We used an implementation based on libSVM [Chang and Lin, 2001]). In a first setting, we applied the classifiers in their original form and recorded the results gained with our performance measures introduced earlier. After that, we used each classifier in combination with our user centric hierarchical approach and again recorded the results.

5.1 Data Sets

We evaluated our algorithm with two datasets. The first is the banksearch dataset [Sinka and Corne, 2002], consisting of 11000 web pages in a 2 level hierarchy (see Fig. 3). The second is a part of the Open Directory [DMOZ, 2006], consisting of 8132 web pages in a hierarchy of depth up to 4 (see Fig. 4). The number of child nodes varies from 2 to 17. We crawled the data in April 2006 and selected the categories presented in Fig. 4. Small subcategories were merged into their parent category.

- **Banking & Finance** (0 docs)
  - Commercial Banks (1000 docs)
  - Building Societies (1000 docs)
  - Insurance Agencies (1000 docs)
- **Programming Languages** (0 docs)
  - Java (1000 docs)
  - C/C++ (1000 docs)
  - Visual Basic (1000 docs)
- **Science** (0 docs)
  - Astronomy (1000 docs)
  - Biology (1000 docs)
- **Sport** (1000 docs)
  - Soccer (1000 docs)
  - Motor Sport (1000 docs)

Figure 3: The Banksearch Dataset

Our two datasets have quite different characteristics. The banksearch dataset has a rather shallow hierarchy and its classes can be separated quite well (in a classification sense). It was created by researchers for evaluation purposes only. On the other hand, the Open Directory data set is far more difficult. The data was structured by different people without any intention to provide a dataset, which is easy to classify (by machine learning methods). In other words, it is truly a "real world" dataset, which makes it clearly more interesting for evaluation from a practical point of view.

5.2 Preprocessing

After parsing the documents, we filtered out terms that occurred in less than five documents, in almost all documents ($> |D| - 5$), stop words, terms with less than four characters, and terms containing numbers. After that, we selected

- **Fitness** (124)
  - Certification (38)
  - Gyms (4)
    - Europe (88)
    - North_America (0)
      - Canada (35)
      - United_States (351)
    - Oceania (32)
  - Personal_Training (86)
  - Pilates_Method (55)
  - Services (31)
- **Society** (0)
  - Activism (131)
    - Anti-Corporation (75)
    - Internet (35)
    - In_Daily_Life (49)
    - Media (50)
      - Culture_Jamming (215)
      - Radio (149)
    - Nonviolence (55)
    - Regional (180)
    - Resources (41)
- **Society** (cont.)
  - Paranormal (144)
    - Bermuda_Triangle (32)
    - Crop_Circles (87)
    - Ghosts (47)
      - Investigators (214)
      - Personal_Pages (47)
      - Places_and_Hauntings (68)
      - Stories (39)
    - Personal_Pages (83)
    - Prophecies (81)
    - Psychic (75)
      - Animals (60)
      - Entertainers (30)
      - ESP (59)
      - Healers (28)
      - Ouija (86)
      - Personal_Pages (30)
      - Readings (444)
      - Teaching (56)
    - UFOs (282)
- **Travel** (26)
  - Guides_and_Directories (115)
  - Image_Galleries (149)
  - Lodging (13)
    - Bed_and_Breakfast (18)
    - Consolidators (65)
    - Directories (19)
    - Home_Exchanges (11)
    - Hospitality_Clubs (15)
    - Hostels (24)
      - Africa (31)
      - Asia (16)
      - Europe (95)
      - North_America (14)
      - Oceania (72)
    - Hotels_and_Motels (26)
    - Vacation_Rentals (73)
  - Preparation (37)
    - Currencies (22)
    - Health (84)
    - Passports_and_Visas (84)
  - Publications (225)
- **Travel** (cont.)
  - Specialty_Travel (246)
    - Adventure_and_Sports (215)
    - Archaeology (41)
    - Arts (126)
    - Backpacking (81)
    - Battlefields (38)
    - Boat_Charters (677)
    - Corporate (130)
    - Cruises (289)
    - Culinary (114)
    - Ecotourism (253)
    - Educational (40)
    - Family (45)
    - Pilgrimage (22)
    - Rail (41)
    - Spas (86)
    - Students (91)
    - Volunteering (137)
  - Transportation (14)
    - Air (259)
    - Car_Rentals (33)
    - Limousines_and_Shuttles (104)

Figure 4: The Open Directory Dataset; the number behind the category name indicates the number of documents directly assigned to this category.

out of these the 100 most distinctive features per class as described in [Borgelt and Nürnberger, 2004]. Each document that had an empty feature vector after this preprocessing was ignored. Each classifier was learned with the same training data. For the banksearch data, we have chosen 300 randomly selected documents from each class, and for the Open Directory data, we have chosen two third of the data in each class. The classifiers were tested with the remainder of the data.

5.3 Results

Tables 1 and 2 summarize our results using the presented evaluation measures. The results of the baseline classifiers are shown in direct comparison to our user centric approach. For each combination of dataset and baseline classifier the results are presented for the optimal setting of the parameter t. As can be seen, the influence on t of the classifier is stronger than the influence of the dataset. This is due to the different qualities of the probability estimates of the two classifiers. In future work, we want to further investigate the influence of the dataset by applying the algorithms to more datasets.

If combining results of the baseline to our user centric approach, it can be seen that our approach generally achieves significant improvements for all performance measures on both data sets. By comparing the performance on the Open Directory data in Table 2 to the results on the banksearch data in Table 1, one can find that the improvements gained by the UCHC method are much higher for the Open Directory data. This was also expected by us as this dataset is a lot more difficult. (This is also proven by the baseline performance results that are much lower for the Open Directory data. E.g., the accuracy of the SVM drops from 93% to 68%.) In specific, the statistics show, e.g., that, if we assume a search strategy as discussed throughout the paper, with UCHC(NB) 29.8% $((\underline{\#n_c}(NB) - \underline{\#n_c}UCHC(NB))/|D = (1391 - 592)/2681)$ more of the data can be retrieved for the Open Directory data, in comparison to 9.3% $((1341-633)/7626)$ for the banksearch data.

As an example of the effect of our user centric approach, the classification tree of documents from the Sport class of the Banksearch dataset is shown in Fig. 5. As can be seen, most documents that had been classified in too specific classes by the pure Naïve Bayes approach (87 documents in *Soccer* and *MS*) were correctly moved up the hierarchy by the UCHC(NB) approach and only 12 documents remained in the too specific classes. Furthermore, 2 out of 25 documents classified in nodes not on the retrieval path had been moved up to the root node. This effect occurs for all classes as can be seen by the $\underline{\#n_c}$ values in Tab. 1. Furthermore, accuracy as well as precision and f-measure could be improved.

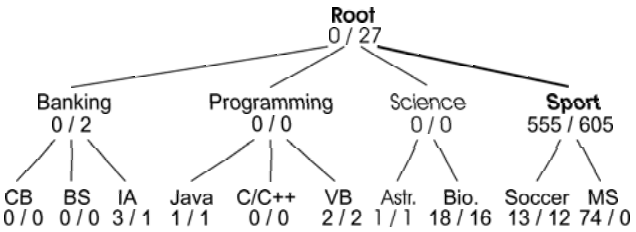

Figure 5: Classification of the Banksearch Sport Documents with Naïve Bayes / UCHC(NB).

Finally, we want to point out that our algorithm aims on generalizing predictions that are possibly wrong because the classifier is uncertain about what the correct class is. That could be, e.g., documents belonging to more than one class or documents, for which a specific class not yet exists. Setting a threshold to generalize wrongly classified documents will therefore almost always also lead to a generalization of correctly classified items, because they also might belong to more than one class. For these documents,

Table 1: Performance Results for the Banksearch Data

	acc_h	$prec_h$	rec_h	f_h	$\#n_c$	$\overline{\#n_c}\,(\overline{ml(n_c)})$	$\underline{\#n_c}\,(\underline{ml(n_c)})$
Naïve Bayes	0.8216	0.8358	0.8215	0.8286	6236	49 (1.0)	1341 (1.38)
UCHC(NB, 20.0)	0.8458	0.9003	0.8461	0.8724	5879	1114 (1.39)	633 (1.20)
SVM	0.9273	0.9278	0.9272	0.9275	7062	16 (1.0)	548 (1.40)
UCHC(SVM, 1.0)	0.9282	0.9319	0.9280	0.9300	7030	84 (1.14)	512 (1.41)

Table 2: Performance Results for the Open Directory Data

	acc_h	$prec_h$	rec_h	f_h	$\#n_c$	$\overline{\#n_c}\,(\overline{ml(n_c)})$	$\underline{\#n_c}\,(\underline{ml(n_c)})$
Naïve Bayes	0.4520	0.5436	0.3056	0.3912	1117	173 (1.26)	1391 (1.98)
UCHC(NB, 26.0)	0.5175	0.7597	0.4096	0.5323	886	1203 (1.98)	592 (1.74)
SVM	0.6847	0.6801	0.5486	0.6073	1687	86 (1.29)	759 (1.64)
UCHC(SVM, 1.5)	0.7154	0.7922	0.6086	0.6884	1518	563 (1.40)	451 (1.71)

it might be just a coincidence that the classification algorithm and the person who created the dataset did the same class assignment. Furthermore, this means (and is shown by the experiments) that our algorithms is especially useful, if the data has a more complex hierarchical structure.

6 Conclusion

In this paper, we presented a user centric approach to hierarchical classification that targets on avoiding misclassifications that would hinder the user to retrieve valuable information. Furthermore, we argued that standard evaluation measures are not suitable to evaluate this kind of setting and propose adaptations to them that take the user behavior into account. We then applied our algorithm to two benchmark datasets and evaluated the results using our proposed measures in comparison to standard classification approaches. The empirical evaluation as presented in the previous section has shown significant improvements of our method over the standard methods, especially on more complex hierarchies. For future work, we consider learning a different threshold parameter t for each class in the hierarchy. Such a class specific threshold could better adjust to class specific differences. However, the available training data for determining the best threshold value is much smaller, which could on the other hand decrease performance again.

References

[Bade and Nürnberger, 2005] Korinna Bade and Andreas Nürnberger. Supporting web search by user specific document categorization: Intelligent bookmarks. In *Proc. of LIT05*, 2005.

[Borgelt and Nürnberger, 2004] Christian Borgelt and Andreas Nürnberger. Fast fuzzy clustering of web page collections. In Mirco Nanni, Michelangelo Ceci, and Marco Gori, editors, *Proc. of PKDD Workshop on Statistical Approaches for Web Mining (SAWM)*, 2004.

[Cai and Hofmann, 2004] L. Cai and T. Hofmann. Hierarchical document categorization w. support vector machines. In *Proc. of 13th ACM Conf. on Inf. and Knowl. Management*, 2004.

[Ceci and Malerba, 2003] M. Ceci and D. Malerba. Hierarchical classification of html documents with webclassii. In *Proc. of 25th Europ. Conf. on Inform. Retrieval*, 2003.

[Cesa-Bianchi *et al.*, 2004] N. Cesa-Bianchi, C. Gentile, A. Tironi, and L.Zaniboni. Incremental algorithms for hierarchical classification. In *Neural Information Processing Systems*, 2004.

[Chang and Lin, 2001] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[Choi and Peng, 2004] B. Choi and X. Peng. Dynamic and hierarchical classification of web pages. *Online Information Review*, 28(2):139–147, 2004.

[DMOZ, 2006] Open directory project, www.dmoz.org, 2006.

[Dumais and Chen, 2000] S. T. Dumais and H. Chen. Hierarchical classification of web content. In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 256–263, 2000.

[Frommholz, 2001] I. Frommholz. Categorizing web documents in hierarchical catalogues. In *Proc. of the European Colloquium on Information Retrieval Research*, 2001.

[Granitzer and Auer, 2005] M. Granitzer and P. Auer. Experiments with hierarchical text classification. In *Proc. of 9th IASTED International Conference on Artifical Intelligence*, 2005.

[Hotho *et al.*, 2005] Andreas Hotho, Andreas Nürnberger, and Gerhard Paaß. A brief survey of text mining. *GLDV-J. for Computational Linguistics and Language Technology*, 20(1):19–62, 2005.

[McCallum *et al.*, 1998] A. K. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 359–367, 1998.

[Sinka and Corne, 2002] M. Sinka and D. Corne. A large benchmark dataset for web document clustering. In *Soft Computing Systems: Design, Management and Applications, Volume 87 of Frontiers in Artificial Intelligence and Applications*, pages 881–890, 2002.

[Sun and Lim, 2001] A. Sun and E. Lim. Hierarchical text classification and evaluation. In *Proc. of the 2001 IEEE International Conference on Data Mining*, pages 521–528, 2001.